

AD-A273 817



TNO Defence Research

TN
Lal

TD

93-30422

①

TDCK RAPPORTENCENTRALE

Frederikkazerne, gebouw 140
v/d Burchlaan 31 MPC 16A
TEL. : 070-3166394/6395
FAX. : (31) 070-3166202
Postbus 90701
2509 LS Den Haag



DTIC

DEC 16 1993

93-30422



93 12 15 04 3

TD 93-27

TNO Defence Research

TNO Physics and Electronics
Laboratory

Oude Waalsdorperweg 63
2597 AK The Hague
P.O. Box 96864
2509 JG The Hague
The Netherlands

Fax +31 70 328 09 61
Phone +31 70 326 42 21

ONGERUBRICEERD

TNO-report
FEL-93-B113

copy nr.

title

A Real-Time Convolution Algorithm and Architecture with
Applications in SAR Processing


author(s):

L.H.J. Bierens

date:

October 1993

TDCK RAPPORTENCENTRALE

Frederikkazerne, gebouw 140
v/d Burchlaan 31 MPC 16A
TEL. : 070-3166394/6395
FAX. : (31) 070-3166202
Postbus 90701
2509 LS Den Haag 

classification

classified by : J.P. van Bezouwen

classification date : October 1993

title : ongerubriceerd

managementuittreksel : ongerubriceerd

abstract : ongerubriceerd

report text : ongerubriceerd

All rights reserved.

No part of this publication may be
reproduced and/or published by print,
photoprint, microfilm or any other means
without the previous written consent of
TNO.

In case this report was drafted on
instructions, the rights and obligations of
contracting parties are subject to either the
'Standard Conditions for Research
Instructions given to TNO' or the relevant
agreement concluded between the
contracting parties.

Submitting the report for inspection to
parties who have a direct interest is
permitted.

TNO

no. of copies : 29

no. of pages : 27 (excluding RDP and distribution list)

no of appendices : -

All information which is classified according to
Dutch regulations shall be treated by the recipient in
the same way as classified information of
corresponding value in his own country. No part of
this information will be disclosed to any party.

The classification designation ONGERUBRICEERD
is equivalent to UNCLASSIFIED.

ONGERUBRICEERD



Netherlands organization for
applied scientific research

TNO Defence Research consists of
the TNO Physics and Electronics Laboratory,
the TNO Prins Maurits Laboratory and the
TNO Institute for Perception



The Standard Conditions for Research Instructions
given to TNO, as filed at the Registry of the District Court
and the Chamber of Commerce in The Hague
shall apply to all instructions given to TNO

DTIC QUALITY INSPECTED 1

MANAGEMENTUITTREKSEL

By	
Distribution /	
Product / Project	
Dist	
A-1	

Titel : Een Real-Time Convolutie Algoritme en Architectuur met Toepassingen in SAR Processing

Auteur(s) : Ir. L.H.J. Bierens

Datum : oktober 1993

Rapport no. : FEL-93-B113

In 1990 is het FEL-TNO in samenwerking met de TU Delft begonnen met het project real-time SAR (RT-SAR), dat wordt gefinancierd door TNO en het Ministerie van Defensie. Het resultaat van het project is de definitie van een high-performance RT-SAR processor voor defensie doeleinden. De RT-SAR processor specificaties zijn gebaseerd op de PHased ARray Universal Sar (PHARUS), die op dit moment door FEL-TNO wordt ontwikkeld.

Het onderzoek dat in dit rapport wordt beschreven heeft betrekking op een essentiële operatie binnen de RT-SAR processing: de range-/azimuthcompressie. Gezien de complexiteit wordt deze operatie uitgevoerd met behulp van fast convolutie- of correlatie-algoritmes.

Het hiervoor ontwikkelde nieuwe algoritme beeldt een lang een-dimensionaal convolutieprobleem af op een kleiner tweedimensionaal convolutieprobleem. Dit wordt bereikt door de lange convolutie te sectioneren in meerdere korte convoluties. De architectuur die hieruit voortvloeit maakt het mogelijk om met standaard FFT processor-elementen een efficiënte implementatie in hardware te realiseren.

De performance met betrekking tot data rate, input data reekslengte en de benodigde hardware is zeer hoog in vergelijking met de standaard gehanteerde configuraties. De architectuur wordt toegelicht aan de hand van een configuratie die geschikt is voor real-time on-board SAR processing voor de PHARUS.

ABSTRACT

In this report we propose an algorithm that maps a large one-dimensional linear convolution problem onto a small two-dimensional linear convolution problem. We show that this property can be derived from a signal flow representation of a sectioned convolution. The introduction of short length FFT processor elements into the signal flow graph results in a structure that can be efficiently implemented in a dedicated hardware architecture. Although the performance of the architecture is not optimal in terms of computational complexity, the performance in terms of data rate, input data sequence lengths and amount of required hardware is high. The architecture design is illustrated with a configuration suitable for real-time on-board airborne synthetic aperture radar (SAR) processing for the PHased ARray Universal Sar (PHARUS), which is currently under development at TNO-FEL.

SAMENVATTING

In dit rapport wordt een algoritme beschreven dat een lang een-dimensionaal convolutieprobleem op een kleiner tweedimensionaal convolutieprobleem afbeeldt. Dit kan worden afgeleid van de signal flow representatie van een gesectioneerd convolutie probleem. De introductie van korte FFT processelementen in de signal flow graph resulteert in een structuur die efficiënt kan worden geïmplementeerd in een dedicated hardware architectuur. De performance met betrekking tot data rate, input data reekslengte en de benodigde hardware is hoog, ondanks dat de performance van de architectuur niet optimaal is wat complexiteit betreft. De architectuur wordt toegelicht aan de hand van een configuratie die geschikt is voor real-time on-board synthetic aperture radar (SAR) processing voor de PHased ARray Universal Sar (PHARUS), die op dit moment door FEL-TNO wordt ontwikkeld.

MANAGEMENTUITTREKSEL	2
ABSTRACT	3
SAMENVATTING	3
1 INTRODUCTION	5
2 THE ONE-DIMENSIONAL CONVOLUTION PROBLEM	8
3 SECTIONING OF LARGE CONVOLUTION PROBLEMS	10
4 SHORT LENGTH FFT PROCESSING	13
5 SIMULATION RESULTS	18
6 AN IMPLEMENTATION EXAMPLE	22
7 SUMMARY	25
REFERENCES	26

1 INTRODUCTION

In 1990 TNO-FEL started a real-time SAR (RT-SAR) processing project in cooperation with the Delft University of Technology [3]. The project is funded by TNO and the Ministry of Defense.

The RT-SAR project is an extension of the SAR activities at TNO-FEL. The RT-SAR processor will be used in combination with the Phased Array Universal SAR (PHARUS), a fully polarimetric SAR, that is currently under development at TNO-FEL.

The defense market increasingly demands high-quality remote sensing images in combination with real-time processing. The advantage of using SAR for military and civil applications is that the quality of a SAR image is independent of weather conditions, clouds, daylight and range (distance between sensor and target). In combination with real-time processing instantaneous anticipation is possible, which is important in crisis situations. Examples of military applications that require real-time processing and high-quality images are surveillance and identification and remotely piloted vehicles (RPV). Moreover, with an RT-SAR processor it is possible to monitor the SAR system, which increases the efficiency of an operational mission.

Increasingly complex digital signal processing (DSP) applications can be performed in real-time with compact hardware due to the developments in DSP hardware and VLSI technology. To serve the defense market in the future it is necessary to keep in touch with these developments. Therefore TNO-FEL has defined RT-SAR processing as an application for the development of dedicated real-time DSP hardware. Moreover, SAR processing is related to applications, such as image processing, inverse SAR (ISAR), interferometry, acoustics, sonar. The RT-SAR technology and experience can be used to design dedicated DSP hardware for these applications.

The main part of the SAR processing consists of convolutions of the data sequences with a reference sequence. Since data sequences and reference sequences can have lengths of $O(10^3)$, real-time SAR processing requires fast convolution algorithms and a dedicated hardware architecture. The

algorithm and architecture that is described in this report are based on the discrete Fourier transform.

The discrete Fourier transform is of great importance in several digital signal processing applications. Especially the calculation of convolution, using fast Fourier transform algorithms, has been studied extensively. Therefore a lot of effort has been devoted to the development of fast convolution algorithms. Most of them are well-known and summarized in e.g. [4, 13]. However, development is still going on in this area, which is shown in recent papers [7, 10].

In general algorithms for fast convolution section a large convolution problem into smaller convolution problems. Well-known examples are overlap-add and overlap-discard methods based on the linearity of the convolution operation. More sophisticated algorithms are based on abstract algebra, such as the Agarwal-Cooley algorithm [2] which is based on the Chinese remainder theorem. The idea is that a one-dimensional cyclic convolution problem of length $N = N_1 N_2$ can be mapped onto a two-dimensional cyclic convolution problem with dimension $N_1 \times N_2$, provided that N_1 and N_2 are relatively prime. Methods to solve this two-dimensional cyclic convolution problem are proposed in [4].

Most of the FFT algorithms emphasize optimum performance in terms of computational complexity or the mapping on (parallel) general purpose computer architectures. However, in many applications, such as real-time radar and acoustical signal processing, performance is also assessed in terms of the amount of hardware and power consumption needed. Moreover, nowadays dedicated VLSI FFT processors for lengths up to 1024 points complex are available as standard components.

In this report we will describe a fast convolution algorithm that can be implemented in a real-time dedicated hardware architecture. The prime requirement is that the hardware architecture must be realized using standard components only, like FFT processors and multipliers. Moreover, the final system must be compact and suitable for on-board applications. We will assume that the data sequences have lengths up to 10^4 , however this can easily be extended.

Our starting point will be the one-dimensional convolution problem. In section 2, we shall describe

convolution using a dependence graph description. In section 3, we show that if the dimension of our convolution algorithm is limited, the convolution problem can be sectioned in time domain using the overlap-add method. This is described by a signal flow graph mapping of the overlap-add convolution algorithm. In section 4, we introduce the short length FFT, which is used to perform short convolution operations within the signal flow graph. The overlap-add convolution algorithm with short length FFTs is recognized as a linear two-dimensional convolution algorithm in frequency domain. The dimensions equal the lengths of the short FFTs. Although not optimal in terms of computational complexity, the structure can be implemented very efficiently in a hardware architecture. In section 5, we give an example of an architecture, using one FFT processor, a complex multiplier, work memory and an adder. The architecture has been simulated using specifications of state-of-the-art standard components to determine the performance in terms of data rate, memory size and maximum input data sequence length. In section 6 the simulated architecture is illustrated with a configuration suitable for real-time on-board airborne SAR processing for the PHased ARray Universal Sar (PHARUS), which is currently under development at TNO-FEL.

2

THE ONE-DIMENSIONAL CONVOLUTION PROBLEM

Let \underline{a} and \underline{x} be two sequences containing discrete time samples defined as $\underline{a} = \{a_n | n = \dots, -1, 0, 1, \dots\}$ and $\underline{x} = \{x_n | n = \dots, -1, 0, 1, \dots\}$. Let $\underline{r} = \{r_n | n = \dots, -1, 0, 1, \dots\}$ be the result of the convolution of \underline{a} and \underline{x} , represented as

$$\underline{r} = \underline{a} * \underline{x} \quad (2.1)$$

then r_n is given as

$$r_n = \sum_{k=-\infty}^{\infty} a_k x_{n-k} \quad (2.2)$$

If \underline{a} and \underline{x} have finite length L_a and L_x , respectively, then \underline{r} has length $L_a + L_x - 1$. If we assume that $a_n = 0$ for $n < 0$ and $n \geq L_a$ and $x_n = 0$ for $n < 0$ and $n \geq L_x$ then r_n is given as

$$r_n = \sum_{k=0}^{L_a-1} a_k x_{n-k} \quad (2.3)$$

The convolution operation can also be described recursively. The recursive form of equation (2.3) is given as

$$r_n^{(k)} = r_n^{(k-1)} + a_k x_{n-k} \quad (2.4)$$

where $n = 0, \dots, L_a + L_x - 2$, $k = 0, \dots, L_a - 1$ and $r_n^{(-1)} = 0$. From equation (2.4) the dependence graph of figure 2.1.a can be derived [9]. Each node in the graph performs a multiplication and an addition (figure 2.1.b). The computational complexity of a length N convolution in time domain is $O(N^2)$.

From a computational point of view it is more efficient to transform the sequences to frequency domain first, where the computational complexity for convolution is much lower, and then perform an inverse transformation. The transformation that is required is known as the discrete Fourier transform (DFT) and can be efficiently computed by the fast Fourier transform (FFT) [11, 12]. Let

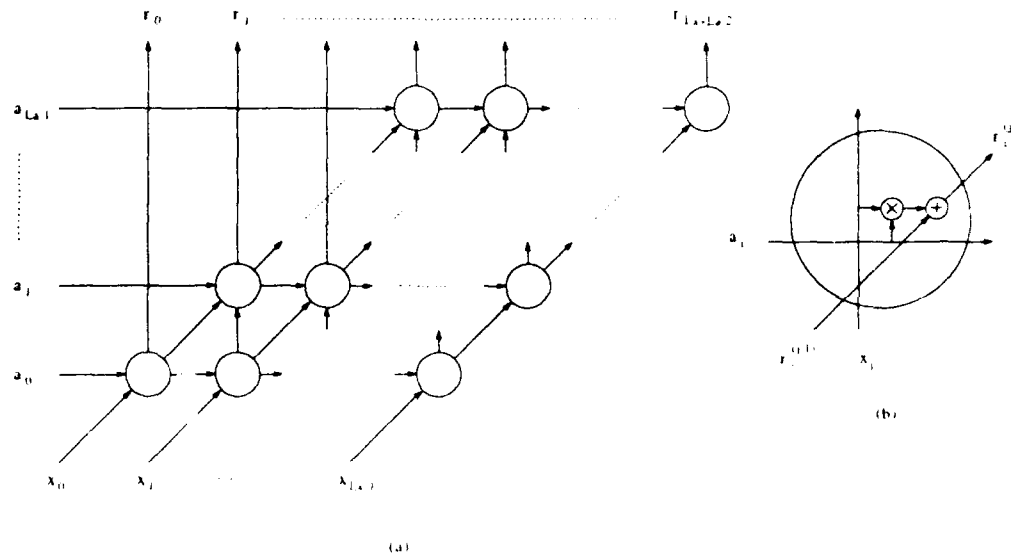


Figure 2.1: Dependence graph for convolution (a) and the elementary operation (b).

$\underline{y} = \{y_n | n = 0, \dots, N-1\}$ be a finite length sequence, then its DFT $\underline{Y} = \{Y_k | k = 0, \dots, N-1\}$ is defined as

$$Y_k = \sum_{n=0}^{N-1} y_n W_N^{nk} \quad (2.5)$$

where $W_N = e^{-j\frac{2\pi}{N}}$. DFT for sequences with length N shall be referred to as N point DFT.

The frequency domain convolution algorithm can be described as follows. Let \underline{a} and \underline{x} be two finite length sequences with lengths L_a and L_x , respectively. Let \underline{A} and \underline{X} be the N point DFTs of \underline{a} and \underline{x} , respectively, with $N = L_a + L_x - 1$. Then $\underline{R} = \{R_k | k = 0, \dots, N-1\}$, with $R_k = A_k X_k$, is the N point DFT of \underline{r} . The computational complexity of convolution in frequency domain is $O(N \log N)$.

3

SECTIONING OF LARGE CONVOLUTION PROBLEMS

In the previous section we have considered the general description of convolution in time domain and in frequency domain. However, when N is too large, which can be the case in many applications, the direct implementations of the time domain or frequency domain algorithms will be inefficient and impractical, especially when dedicated architectures are used. Therefore we can use *overlap-add* or *overlap-discard* methods to section one or both sequences in smaller subsequences. In this paper we shall focus on the overlap-add method. For the overlap-discard method we refer to [11, 12].

Let \underline{a} and \underline{x} be defined as in the previous section and assume that we have some convolution architecture for input sequences of length L only (either in time domain or frequency domain), with $L \ll L_a$ and $L \ll L_x$. Then we can section our convolution problem as follows. Let the subsequences \underline{a}_{la} and \underline{x}_{lx} of \underline{a} and \underline{x} , respectively, be defined as

$$\begin{aligned} \underline{a}_{la} &= \begin{cases} \{a_{l+L+l} \mid l = 0, \dots, L-1\} & \text{if } 0 \leq la < L'_a, \quad L'_a = \lceil \frac{L_a}{L} \rceil \\ 0 & \text{otherwise} \end{cases} \\ \underline{x}_{lx} &= \begin{cases} \{x_{l+L+l} \mid l = 0, \dots, L-1\} & \text{if } 0 \leq lx < L'_x, \quad L'_x = \lceil \frac{L_x}{L} \rceil \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Substitute

$$k \triangleq k1 \cdot L + k2, \quad k2 = 0, \dots, L-1$$

$$n \triangleq n1 \cdot L + n2, \quad n2 = 0, \dots, L-1$$

(3.1)

in equation (2.3), hence

$$\begin{aligned} r_{n1 \cdot L + n2} &= \sum_{k1=0}^{L'_a-1} \sum_{k2=0}^{L-1} a_{k1 \cdot L + k2} x_{n1 \cdot L + n2 - (k1 \cdot L + k2)} \\ &= \sum_{k1=0}^{L'_a-1} \sum_{k2=0}^{L-1} a_{k1 \cdot L + k2} x_{(n1-k1)L + n2 - k2} \end{aligned} \quad (3.2)$$

The interpretation in the dependence graph is as follows. Consider the dependence graph in figure 3.1, where we have assumed that L_a and L_x are multiples of L . Obviously equation (3.2)

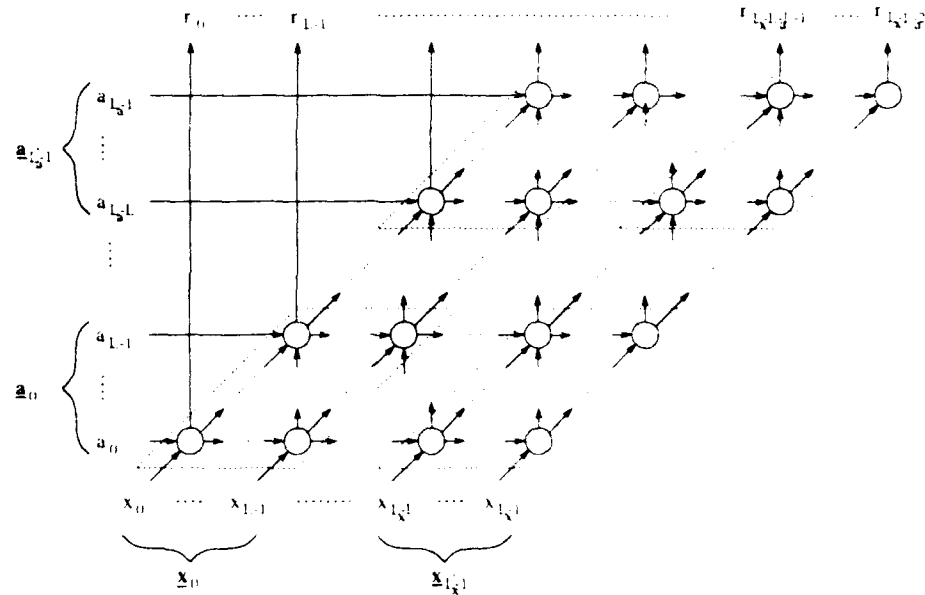


Figure 3.1: Dependence graph for convolution with clustering of $L \times L$ nodes.

can be interpreted as the clustering of $L \times L$ nodes. The arithmetic operation within a node is recognized as a convolution of two subsequences of length L .

Now we can derive a signal flow graph [9] of the dependence graph, where we assume that the processing element performs the convolution operation within one cluster, see figure 3.2.a. The delay operator D_{L-1} represents a delay of $L - 1$ samples. The processing element is shown in figure 3.2.b.

Observe that the computational complexity of time domain convolution with overlap-add has not changed compared to the time domain convolution without overlap-add. To determine the computational complexity of frequency domain convolution with overlap-add we assume that the input sequence length is $\mathcal{O}(N)$ and the section length is $\mathcal{O}(M)$, with $M \ll N$. Then the computational complexity is given as $\mathcal{O}(\frac{N^2}{M} \log M)$. Notice that overlap-add results in an increase of computational complexity from $\mathcal{O}(N \log N)$ to $\mathcal{O}(\frac{N^2}{M} \log M)$.

One stage of the overlap-add method we did not mention. The subsequences \underline{r}_r , $r = 0, \dots, L'_a + L'_r - 2$ must be added. Since each of the subsequences has been delayed properly within the signal

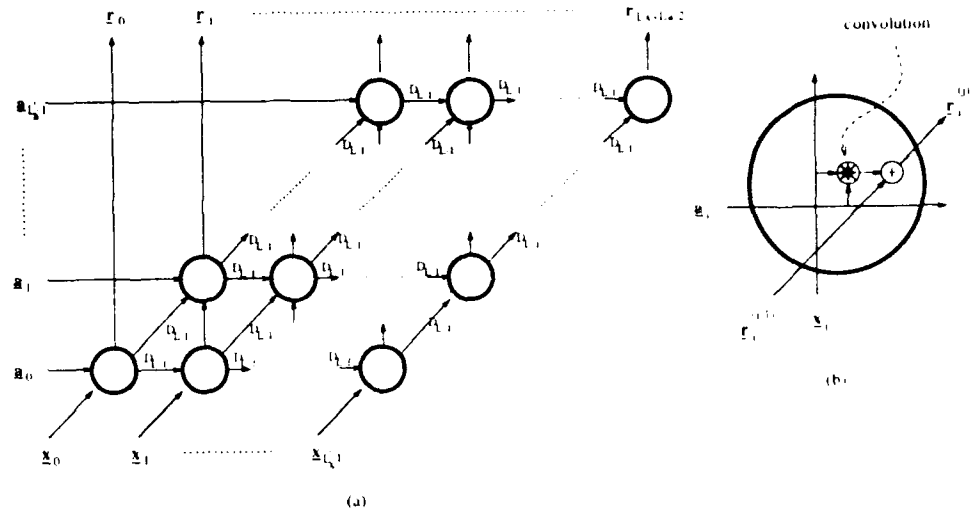


Figure 3.2: Signal flow graph derived from the dependence graph (a) and the processing element (b).

flow graph this can be done straight-forward

$$\underline{r} = \sum_{l=0}^{L'_n + L'_t - 2} \underline{r}_{(l)} \quad (3.3)$$

The additional computational complexity is negligible.

4

SHORT LENGTH FFT PROCESSING

The question that now arises, is how to embed the FFT within the overlap-add structure such that the computational complexity decreases. The straight-forward way (as is often implemented in hardware architectures) is to perform the convolution within a processing element in frequency domain. This results to a processing element as shown in figure 4.1.

Each subsequence \underline{a}_{l_n} and \underline{x}_{l_x} is transformed L'_x times and L'_n times, respectively. However, it is more efficient to make use of the linearity of the FFT. Then we can do one transformation per subsequence at the input side. In the same way we can do one inverse transformation per subsequence at the output side. This results in the signal flow graph in figure 4.2.a, which is equivalent to the signal flow graph in figure 3.2.a. The arithmetic operation within a processing element is now only an element-wise multiplication in stead of a convolution. Notice that we perform the overlap-add in frequency domain.

If we analyze the structure of the signal flow graph of figure 4.2 we recognize that the kernel has exactly the same structure as the dependence graph (except for the delay operators). Moreover, since the multiplication operator within the processing element operates element-wise, the complete kernel as a whole operates element-wise.

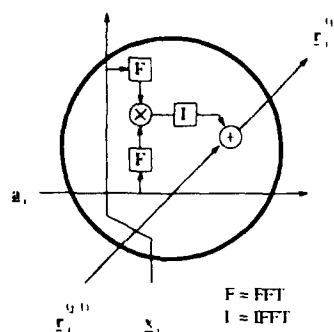


Figure 4.1: Processing element of the signal flow graph. The convolution is performed in frequency domain.

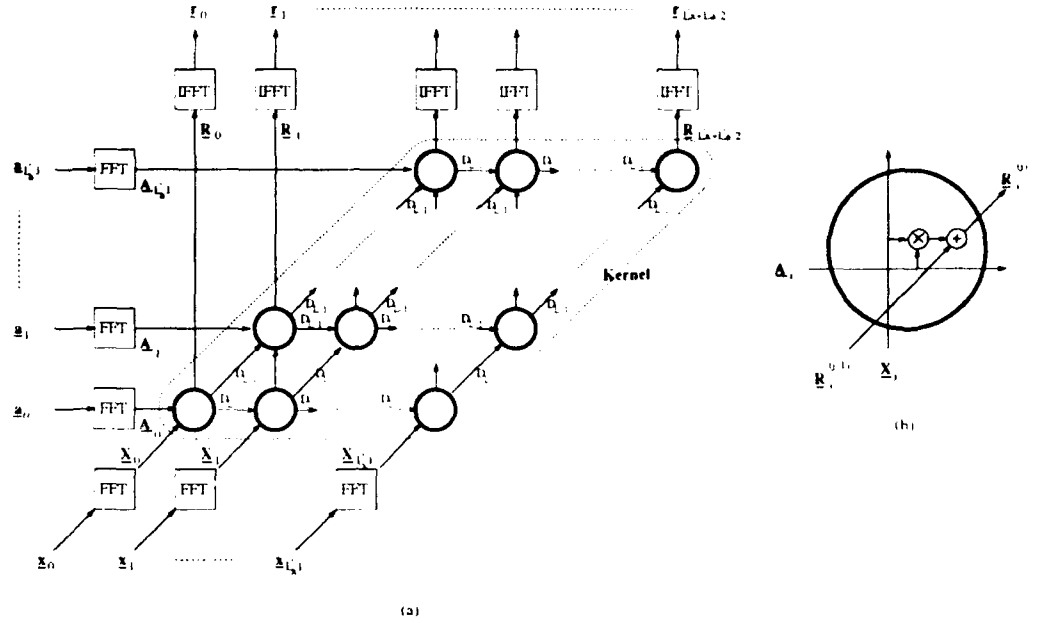


Figure 4.2: Signal flow graph of the overlap-add in frequency domain (a) and the processing element (b).

Let the sequences \underline{A}_{l_a} , \underline{X}_{l_x} and \underline{R}_{l_r} be the FFTs of the subsequences \underline{a}_{l_a} , \underline{x}_{l_x} and \underline{r}_{l_r} , respectively (see figure 4.2.a), with the lengths of \underline{A}_{l_a} , \underline{X}_{l_x} and \underline{R}_{l_r} at least $2L - 1$. Let $A_{l_a,k}$, $X_{l_x,k}$ and $R_{l_r,k}$, $k = 0, 1, \dots$ be the elements of \underline{A}_{l_a} , \underline{X}_{l_x} and \underline{R}_{l_r} , respectively.

The kernel can be represented as

$$\underline{R}_{l_r} = \sum_{l=0}^{L'_a-1} \underline{A}_l \otimes \underline{X}_{l_r-l} \quad (4.1)$$

where \otimes is the element-wise multiplication, and $\underline{X}_{l_r} \triangleq 0$ if $l_r < 0$ and $l_r \geq L'_x$. Equation (4.1) is equivalent with

$$\{R_{l_r,k} | k = 0, 1, \dots\} = \left\{ \sum_{l=0}^{L'_a-1} A_{l,k} X_{l_r-l,k} | k = 0, 1, \dots \right\} \quad (4.2)$$

Obviously, equation (4.1) represents multiple one-dimensional convolutions, as in equation (4.2). The k^{th} convolution sequences are given as $\{A_{l_a,k} | l_a = 0, \dots, L'_a - 1\}$ and $\{X_{l_x,k} | l_x = 0, \dots, L'_x - 1\}$ and the result is given as $\{R_{l_r,k} | l_r = 0, \dots, L'_a + L'_x - 2\}$. Then we can

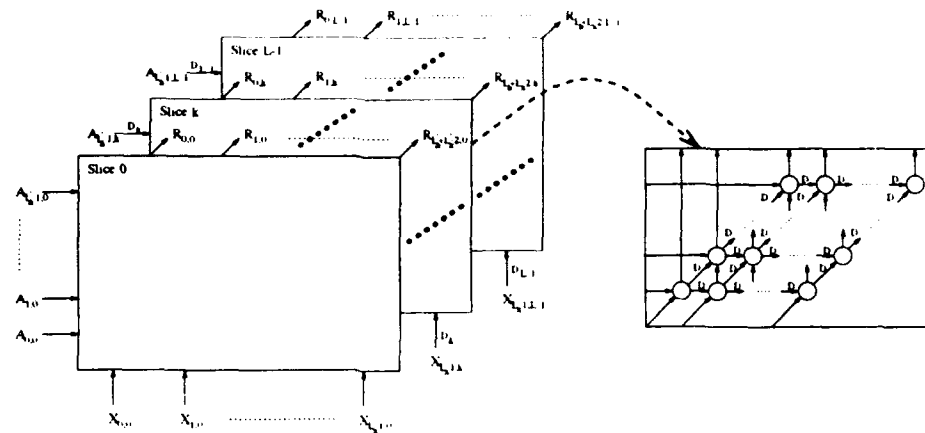


Figure 4.3: Kernel of the signal flow graph of the overlap-add in frequency domain and the processing element. The slices represent signal flow graph of a convolution.

draw the kernel of the signal flow graph in figure 4.2 as in figure 4.3, where each k^{th} slice represents the k^{th} convolution.

Observe that the kernel of the signal flow graph can be expressed as $L - 1$ convolution operations in time domain. Then the next step would be, logically, to perform the convolution operation in frequency domain. The equivalent frequency domain description of one slice is shown in figure 4.4. The FFT lengths must be at least $L'_q + L'_x - 1$. The delay operations have been omitted since they are implied by the geometry. Figure 4.5 gives the complete three-dimensional signal flow graph of the convolver.

To determine the complexity we shall again assume the length of the input sequences and the section length $\mathcal{O}(N)$ and $\mathcal{O}(M)$, respectively. The computational complexity of the FFTs of the subsequences is $\mathcal{O}(N \log M)$ and the computational complexity of the kernel FFTs is $\mathcal{O}(N \log \frac{N}{M})$. Hence the computational complexity of the complete convolution algorithm as shown in figure 4.5 is given as $\mathcal{O}(N \log M + N \log \frac{N}{M}) = \mathcal{O}(N \log N)$.

The three-dimensional structure of the algorithm in figure 4.5 implies that we are doing some two-dimensional signal processing. In fact, the overlap-add method is a mapping of a one-dimensional

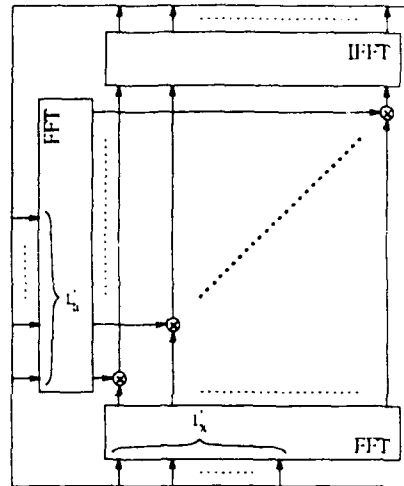


Figure 4.4: The frequency domain description of one slice from the kernel of the signal flow graph of the convolution in frequency domain with overlap-add.

convolution problem to a two-dimensional convolution problem with the same complexity. Consider equation (3.2), and let

$$a_{l_a,l} \triangleq a_{l_a-L+l} \quad x_{l_x,l} \triangleq x_{l_x-L+l} \quad r_{n1,n2} \triangleq r_{n1-L+n2}$$

then equation (3.2) becomes

$$r_{n1,n2} = \sum_{k1=0}^{L'_a-1} \sum_{k2=0}^{L'_x-1} a_{k1,k2} x_{n1-k1,n2-k2} \quad (4.3)$$

which is the definition of a two-dimensional linear convolution.

This result has been derived in [1] and developed further in [2, 5]. In [4, 13] the algorithm that solves a one-dimensional convolution problem by mapping it onto a two-dimensional convolution problem is summarized, referred to as the Agarwal-Cooley convolution algorithm. One should notice that the Agarwal-Cooley convolution algorithm is only defined for $N_1 N_2$ -point one-dimensional cyclic convolutions, with N_1 and N_2 relatively prime. The algorithm that we propose solves a linear convolution of sequences with arbitrary length with a two-dimensional linear convolution of dimensions $2L - 1 \times L'_a + L'_x - 1$. The only requirement is that we need two FFTs of at least $2L - 1$ points and $L'_a + L'_x - 1$ points, respectively.

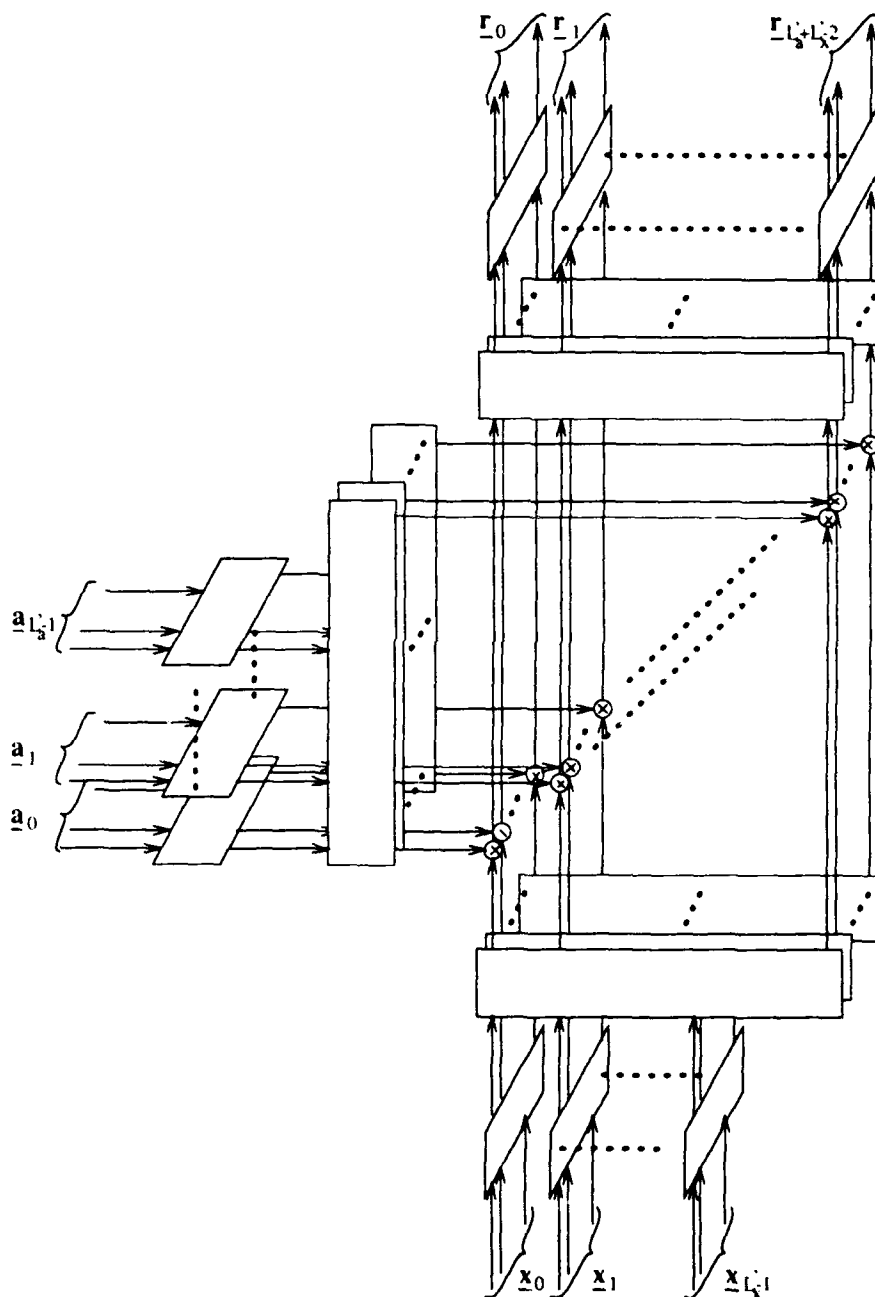


Figure 4.5: Three-dimensional signal flow graph of a convolution algorithm that consists of FFTs and multiplications. In stead of long FFTs in one-dimension short FFTs are used in two-dimensions. The polygons represent the FFTs.

5 SIMULATION RESULTS

Algorithm 1 gives a straight-forward representation of the convolution algorithm. It is obvious that this is not optimal in terms of computational complexity. However, the structure of the algorithm makes it very suitable for implementation in dedicated architectures with short length FFT processor elements.

Algorithm 1 : Convolution Algorithm

```

STAGE 1 for  $k = 0 : L'_a - 1$ 
     $\{A_{k,0}, \dots, A_{k,2(L-1)}\} \xrightarrow{FFT} \{a_{k,0}, \dots, a_{k,L+L-1}\}$ 
end
for  $k = 0 : L'_a - 1$ 
     $\{X_{k,0}, \dots, X_{k,2(L-1)}\} \xrightarrow{IFFT} \{x_{k,0}, \dots, x_{k,L+L-1}\}$ 
end
STAGE 2 for  $l = 0 : 2(L-1)$ 
     $\{A'_{0,l}, \dots, A'_{L'_a+L'_x-2,l}\} \xrightarrow{FFT} \{A_{0,l}, \dots, A_{L'_a-1,l}\}$ 
end
for  $l = 0 : 2(L-1)$ 
     $\{X'_{0,l}, \dots, X'_{L'_a+L'_x-2,l}\} \xrightarrow{FFT} \{X_{0,l}, \dots, X_{L'_a-1,l}\}$ 
end
STAGE 3 for  $l = 0 : 2(L-1)$ 
    for  $k = 0 : L'_a + L'_x - 2$ 
         $R'_{k,l} \leftarrow A'_{k,l} \times X'_{k,l}$ 
    end
     $\{R'_{0,l}, \dots, R'_{L'_a+L'_x-2,l}\} \xrightarrow{IFFT} \{R'_{0,l}, \dots, R'_{L'_a+L'_x-2,l}\}$ 
end
STAGE 4 for  $k = 0 : L'_a + L'_x - 2$ 
     $\{r'_{k,0}, \dots, r'_{k,2(L-1)}\} \xrightarrow{IFFT} \{R_{k,0}, \dots, R_{k,2(L-1)}\}$ 
end
STAGE 5 for  $k = 0 : L'_a + L'_x - 2$ 
    for  $l = 0 : L - 1$ 
         $r_{k,L+l} \leftarrow r'_{k,l} + r'_{k-1,L+l}$ 
    end
end

```

To illustrate this we will give an example of an efficient hardware architecture. We assume that we have a complex multiplier and an FFT processor with two modes: a $2L$ point mode and an $L'_a + L'_x$ point mode. Furthermore we have two work memories and some add and delay hardware. The architecture is shown in figure 5.1.

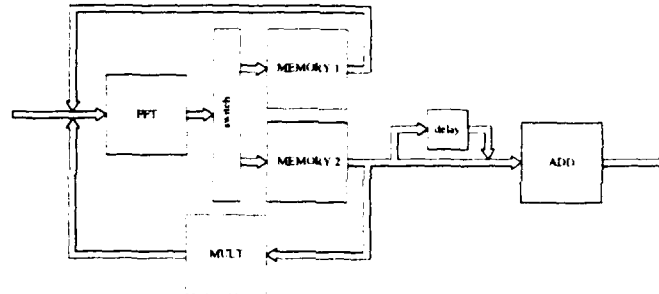


Figure 5.1: An example of a hardware architecture.

The algorithm has been divided in 5 stages (see algorithm 1). Each stage has its own functionality, as is shown in table 5.1. The architecture computes the stages 1 to 5 sequentially. In figure 5.2 the data flow for each stage is shown.

To determine the data rate of the convolution hardware architecture we will assume that $L_n < L_x$. We further assume that the storage capacity of memory 1 is at least $2L(L'_n + L'_x)$ samples and the storage capacity of memory 2 is at least $4L(L'_n + L'_x)$ samples. The data rate is defined as

$$\text{data rate} = \frac{\text{no. of data samples to be processed}}{\text{processing time}}$$

Furthermore we define the following parameters:

- D_{f1} : the data rate of the FFT processor in $2L$ point mode
- D_{f2} : the data rate of the FFT processor in $L'_n + L'_x$ point mode

Stage	In	Process	FFT mode (points)	Out	No. of data samples
1	input	FFT	$2L$	mem. 1	$2L(L'_n + L'_x)$
2	mem. 1	FFT	$L'_n + L'_x$	mem. 2	$4L(L'_n + L'_x)$
3	mem. 2	multiply + IFFT (in pipeline)	$L' + L'_x$	mem. 1	$2L(L'_n + L'_x)$
4	mem. 1	IFFT	$2L$	mem. 2	$2L(L'_n + L'_x)$
5	mem. 2	delay + addition	-	output	$L(L'_n + L'_x)$

Table 5.1: Functional description of each stage.

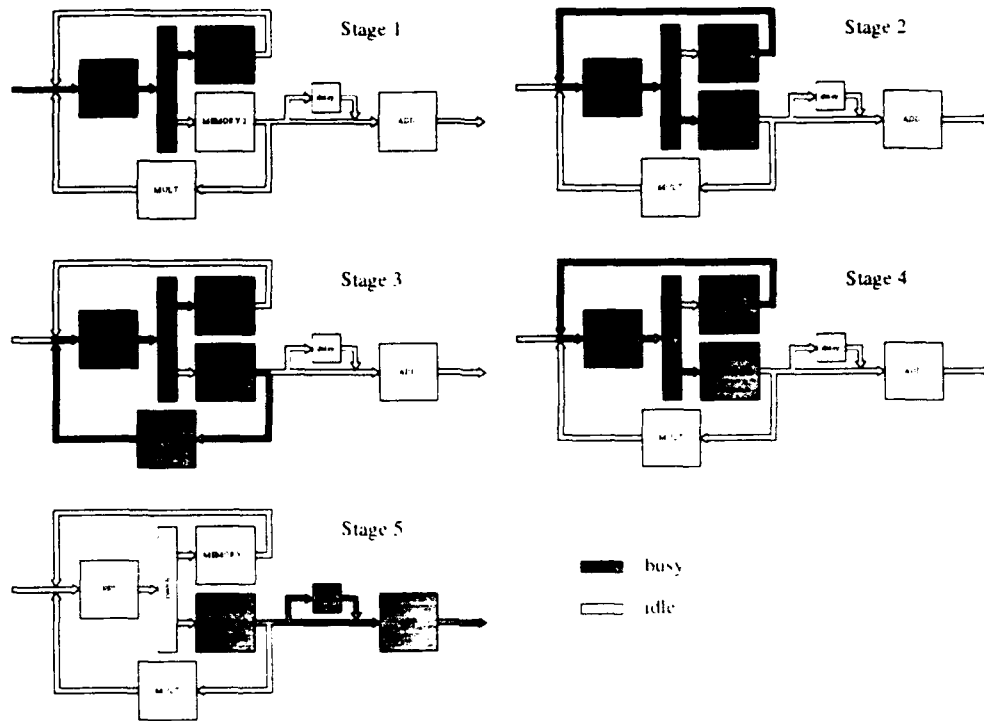


Figure 5.2: Data flow of each stage. Observe that stage 5 can be done in parallel with stage 1.

- D_m : the data rate of the complex multiplier
- P_f : no. of FFT processors in parallel
- P_m : no. of complex multipliers processors in parallel
- $T_i, i = 1, \dots, 4$: processing time of stage i

Then we have

$$T_1 = \frac{2L(L'_0 + L'_1)}{P_f D_{f1}} \quad T_2 = \frac{4L(L'_0 + L'_1)}{P_f D_{f2}}$$

$$T_3 = \frac{2L(L'_0 + L'_1)}{\min\{P_m D_m, P_f D_{f2}\}} \quad T_4 = \frac{2L(L'_0 + L'_1)}{P_f D_{f1}}$$

From figure 5.2 we see that stage 5 can be done in parallel with stage 1. This means the architecture is ready for the next convolution job when stage 4 is finished, so it is not relevant to define T_5 . The data rate D of the architecture is given as

$$D = \frac{L_r}{\sum_{i=1}^4 T_i} \quad (5.1)$$

	L	$L'_a + L'_i$	D ($\times L_i P_{f_i}$)	L_i (maximum)	memory capacity	
					Mem 1	Mem 2
1	8	16	$2 \cdot 10^4$	$128 - L_a$	256	512
2	8	64	$5 \cdot 10^4$	$512 - L_a$	1K	2K
3	8	256	$1 \cdot 10^5$	$2K - L_a$	4K	8K
4	8	1K	$2 \cdot 10^5$	$8K - L_a$	16K	32K
5	32	16	$5 \cdot 10^4$	$512 - L_a$	1K	2K
6	32	64	$1 \cdot 10^5$	$2K - L_a$	4K	8K
7	32	256	$3 \cdot 10^5$	$8K - L_a$	16K	32K
8	32	1K	40	$32K - L_a$	64K	128K
9	128	16	$1 \cdot 10^5$	$2K - L_a$	4K	8K
10	128	64	$3 \cdot 10^5$	$8K - L_a$	16K	32K
11	128	256	80	$32K - L_a$	64K	128K
12	128	1K	10	$131K - L_a$	262K	524K
13	512	16	$1 \cdot 10^5$	$8K - L_a$	16K	32K
14	512	64	40	$32K - L_a$	64K	128K
15	512	256	9	$131K - L_a$	262K	524K
16	512	1K	2	$524K - L_a$	1M	2M

Table 5.2: Performance figures and memory requirements for an architecture with a single-chip state-of-the-art FFT processor. L , $L'_a + L'_i$, L_i and the memory capacity are represented by no. of complex samples. The data rate D is represented by the no. of complex samples per second.

If we require $P_m D_m \leq P_f D_{f2}$ then equation (5.1) becomes

$$D = \frac{L_i}{2L(L'_a + L'_i)} \frac{P_f}{(2D_{f1}^{-1} + 3D_{f2}^{-1})} \quad (5.2)$$

To illustrate the performance of an implementation in hardware with standard components we have simulated the architecture using the specifications of a single-chip state-of-the-art FFT processor. The FFT processor has 4 complex modes: 16 point, 64 point, 256 point and 1024 point. The data rates for these modes are $D_{f1}(16) = D_{f1}(64) = D_{f1}(256) \approx 25$ MSamples/sec and $D_{f1}(1024) \approx 10$ MSamples/sec, $i = 1, 2$. The performance figures and memory requirement are listed in table 5.2. Observe that the data rate increases linearly with the number of FFT processors in parallel.

6 AN IMPLEMENTATION EXAMPLE

In this section we will describe a configuration of a real-time processor architecture for the Phased Array Universal Synthetic Aperture Radar (PHARUS). PHARUS is a phased array polarimetric SAR that is currently under development at TNO Physics and Electronics Laboratory. For detailed specifications we refer to e.g. [6, 8]. We will restrict ourselves to the specifications which are needed for our configuration and we will not consider phase error correction. The specifications¹ are as follows:

- The configuration is determined for one polarimetric direction.
- The number of looks is 4 with 50% overlap.
- The resolution is 4 m.
- The minimum range R_{min} is 7 Km.
- The maximum range R_{max} is 16 Km.
- The sample frequency in range f_r is 100 MHz.
- The pulse length τ_r is 19.7 μ sec.
- The sample frequency in azimuth f_a is 292 Hz.
- The aperture length τ_a is maximally 4.41 sec.

Range Compression

From the sample frequency in range we can determine the distance between the slant range dr as

$$dr = \frac{c}{2f_r} \quad (6.1)$$

¹The specifications have been obtained from PHARUS document no. PH9252FEL.

This implies that the number of range samples N_r is given as

$$N_r = \frac{R_{max} - R_{min}}{dr} = 6 \text{ Ksamples}$$

The processing of the range samples must be performed within $\frac{1}{f_a}$ seconds, which gives the required data rate D_r of the range compression as

$$D_r = N_r f_a \approx 1.75 \text{ Msamples/sec}$$

The number of samples of the reference sequence M_r is given as

$$M_r = \tau_r f_r \approx 2 \text{ Ksamples}$$

If we compare this with table 5.2 we see the requirements are met with the 7th or 10th row, with $P_f = 1$. So we can perform real-time range compression for PHARUS with a configuration with one FFT processor chip in 64 and 256 point mode or 256 and 64 point mode, respectively. For both configurations we need work memories of 16 Ksamples complex and 32 Ksamples complex. The data rate for this configuration is 1.8 Msamples/sec.

Azimuth Compression

The maximum number of samples per aperture M_a is given as

$$M_a = \tau_a f_a \approx 1.3 \text{ Ksamples}$$

Since we have 4 looks with 50% overlap the maximum number of aperture samples per look M'_a is given as

$$M'_a = 520 \text{ samples}$$

Let N_a be the number of azimuth samples per batch that is processed. Then, within $\frac{N_a}{f_a}$ seconds, 4 (the number of looks) times N_r azimuth lines of length N_a must be processed (i.e. convolved with a length M'_a reference sequence). The required data rate D_a of the azimuth compression is thus given as

$$D_a = 4 N_r f_a \approx 7 \text{ Msamples/sec}$$

If we choose $N_a = 1.5$ Ksamples then the requirements are met with the 3th or 6th row of table 5.2, with $P_f = 5$. So we can perform real-time azimuth compression for PHARUS with a configuration with five FFT processor chips in 16 and 256 point mode or 64 and 64 point mode, respectively. For both configurations we need work memories of 4 Ksamples complex and 8 Ksamples complex. The data rate for this configuration is 7.5 Msamples/sec. However, if we choose $N_a = 7.5$ Ksamples then the requirements are met with the 7th or 10th row, with $P_f = 4$. In this case we need four FFT processor chips, but the data sequences get longer, which is not always desirable, and the required work memory sizes increase to 16 Ksamples complex and 32 Ksamples complex.

7 SUMMARY

In this paper we described the mapping of a one-dimensional large linear convolution problem onto a small linear two-dimensional convolution problem. A dependence graph and signal flow graph representation has been used to derive this property. By introducing the short length *FFT* into the signal flow graph representation we obtained a three-dimensional signal flow graph with short length *FFTs* and multiplications only. Although this algorithm is not optimum in terms of computational complexity, it can be efficiently mapped onto a hardware architecture based on standard components. The simulation results showed that the architecture performs very well in terms of system data rate, input data sequence length and amount of required hardware. The architecture is illustrated with a configuration suitable for real-time on-board airborne SAR processing for the *PHased ARray Universal Sar (PHARUS)*. This "quick-look" hardware implementation will be subject for further work in the near future.

REFERENCES

- [1] R.C. Agarwal and C.S. Burrus.
Fast one-dimensional digital convolution by multidimensional techniques.
IEEE-ASSP, ASSP-22(1):1-10, February 1974.
- [2] R.C. Agarwal and J.W. Cooley.
New algorithms for digital convolution.
IEEE-ASSP, ASSP-25(5):392-410, October 1977.
- [3] L.H.J. Bierens.
A real-time synthetic aperture radar processor: Introduction and project description.
Technical Report FEL-91-B295, TNO Physics and Electronics Laboratory, 1991.
- [4] R.E. Blahut.
Fast Algorithms for Digital Signal Processing.
Addison-Wesley, 1985.
- [5] C.S. Burrus.
Index mapping for multidimensional formulation of the DFT and convolution.
IEEE-ASSP, ASSP-25(3):239-242, June 1977.
- [6] P. Hoogenboom et al.
Definition study PHARUS: final report.
Technical Report FEL-91-A375, TNO Physics and Electronics Laboratory, 1991.
- [7] J. Granata, M. Connor, and R. Tolmieri.
Recursive fast algorithms and the role of the tensor product.
IEEE-ASSP, ASSP-40(12):2921-2930, December 1992.
- [8] P. Hoogenboom, P. Snoeij, P.J. Koomen, and H. Pouwels.
The PHARUS project, results of the definition study including the SAR testbed PHARS.
IEEE Transactions on Geoscience and Remote Sensing, 30(4):723-735, July 1992.

- [9] S.Y. Kung.
VLSI Array Processors.
Prentice-Hall, Inc., 1988.
- [10] C. Lu, J.W. Cooley, and R. Tolimieri.
FFT algorithms for prime transform sizes and their implementations on VAX, IBM3090VF,
and IBM RS/6000.
IEEE-ASSP, ASSP-41(2):638-648, February 1993.
- [11] A.L. Oppenheim and R.W. Schaffer.
Digital Signal Processing.
Prentice-Hall, Inc., 1975.
- [12] L.R. Rabiner and B. Gold.
Theory and Application of Digital Signal Processing.
Prentice-Hall, Inc., 1975.
- [13] R. Tolimieri, M. An, and C. Lu.
Algorithms for Discrete Fourier Transforms and Convolution.
Springer-Verlag, 1989.



J.P. van Bezouwen
(Group leader)



L.H.J. Bierens
(Author)

REPORT DOCUMENTATION PAGE

(MOD-NL)

1. DEFENSE REPORT NUMBER (MOD-NL) TD93-2788	2. RECIPIENT'S ACCESSION NUMBER	3. PERFORMING ORGANIZATION REPORT NUMBER FEL-93-B113
4. PROJECT/TASK/WORK UNIT NO. 22458	5. CONTRACT NUMBER	6. REPORT DATE OCTOBER 1993
7. NUMBER OF PAGES 27 (EXCL. RDP + DISTRIBUTION LIST)	8. NUMBER OF REFERENCES 13	9. TYPE OF REPORT AND DATES COVERED
10. TITLE AND SUBTITLE A REAL-TIME CONVOLUTION ALGORITHM AND ARCHITECTURE WITH APPLICATIONS IN SAR PROCESSING		
11. AUTHOR(S) L.H.J. BIERENS		
12. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) TNO PHYSICS AND ELECTRONICS LABORATORY, P.O. BOX 96864, 2509 JG THE HAGUE OUDERWAALSDORPERWEG 63, THE HAGUE, THE NETHERLANDS		
13. SPONSORING/MONITORING AGENCY NAME(S) TNO PHYSICS AND ELECTRONICS LABORATORY		
14. SUPPLEMENTARY NOTES THE CLASSIFICATION DESIGNATION ONGERUBRICEERD IS EQUIVALENT TO UNCLASSIFIED.		
15. ABSTRACT (MAXIMUM 200 WORDS, 1044 POSITIONS) IN THIS REPORT WE PROPOSE AN ALGORITHM THAT MAPS A LARGE ONE-DIMENSIONAL LINEAR CONVOLUTION PROBLEM ONTO A SMALL TWO-DIMENSIONAL LINEAR CONVOLUTION PROBLEM. WE SHOW THAT THIS PROPERTY CAN BE DERIVED FROM A SIGNAL FLOW REPRESENTATION OF A SECTIONED CONVOLUTION. THE INTRODUCTION OF SHORT LENGTH FFT PROCESSOR ELEMENTS INTO THE SIGNAL FLOW GRAPH RESULTS IN A STRUCTURE THAT CAN BE EFFICIENTLY IMPLEMENTED IN A DEDICATED HARDWARE ARCHITECTURE. ALTHOUGH THE PERFORMANCE OF THE ARCHITECTURE IS NOT OPTIMAL IN TERMS OF COMPUTATIONAL COMPLEXITY, THE PERFORMANCE IN TERMS OF DATA RATE, INPUT DATA SEQUENCE LENGTHS AND AMOUNT OF REQUIRED HARDWARE IS HIGH. THE ARCHITECTURE DESIGN IS ILLUSTRATED WITH A CONFIGURATION SUITABLE FOR REAL-TIME ON-BOARD AIRBORNE SYNTHETIC APERTURE RADAR (SAR) PROCESSING FOR THE PHASED ARRAY UNIVERSAL SAR (PHARUS), WHICH IS CURRENTLY UNDER DEVELOPMENT AT TNO-FEL.		
16. DESCRIPTORS REAL-TIME COMPUTATION SYNTHETIC APERTURE RADAR DATA PROCESSING SIGNAL PROCESSING ALGORITHM		IDENTIFIERS DIGITAL SIGNAL PROCESSING FAST FOURIER TRANSFORM
17a. SECURITY CLASSIFICATION (OF REPORT) ONGERUBRICEERD	17b. SECURITY CLASSIFICATION (OF PAGE) ONGERUBRICEERD	17c. SECURITY CLASSIFICATION (OF ABSTRACT) ONGERUBRICEERD
18. DISTRIBUTION/AVAILABILITY STATEMENT UNLIMITED		17d. SECURITY CLASSIFICATION (OF TITLES) ONGERUBRICEERD